AF/S JFW

PATENT
Customer No. 22,852
Attorney Docket No. 06502.0323

BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

In re Application of:	)	
	)	
Thomas V. JOHNSON et al.	)	Group Art Unit: 2161
	)	
Application No.: 09/895,077	)	Examiner: Brian D. Goddard
	)	
Filed: July 2, 2001	)	
	)	
For: METHODS AND SYSTEM FOR EFFICIENT ASSOCIATION TRAVERSALS	)	Confirmation No.: 9092

Mail Stop Appeal Brief--Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir:

TRANSMITTAL OF APPEAL BRIEF (37 C.F.R. 41.37)

Transmitted herewith is the APPEAL BRIEF in this application with respect to the Notice of Appeal filed on August 5, 2005.

This application is on behalf of

☐ Small Entity ☒ Large Entity

Pursuant to 37 C.F.R. 41.20(b)(2), the fee for filing the Appeal Brief is:

☐ $250.00 (Small Entity)

☒ $500.00 (Large Entity)

TOTAL FEE DUE:

Appeal Brief Fee $500.00

☒ Enclosed is a check for $500.00 to cover the above fees.

PETITION FOR EXTENSION. If any extension of time is necessary for the filing of this Appeal Brief, and such extension has not otherwise been requested, such an extension is hereby requested, and the Commissioner is authorized to charge necessary fees for such an extension to our Deposit Account No. 06-0916. A duplicate copy of this paper is enclosed for use in charging the deposit account.

FINNEGAN, HENDERSON, FARABOW,
GARRETT & DUNNER, L.L.P.

Dated: October 5, 2005

By: _____
John M. Mulcahy
Reg. No. 55,940

PATENT
Customer No. 22,852
Attorney Docket No. 6502.0323-00000

BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

In re Application of:	)	
	)	
Thomas V. JOHNSON et al.	)	Group Art Unit: 2161
	)	
Application No.: 09/895,077	)	Examiner: Brian D. Goddard
	)	
Filed: July 2, 2001	)	
	)	
For: METHODS AND SYSTEM FOR EFFICIENT ASSOCIATION TRAVERSALS	)	Confirmation No.: 9092

Mail Stop Appeal Brief-Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir:

APPEAL BRIEF UNDER BOARD RULE § 41.37

In support of the Notice of Appeal filed August 5, 2005, and further to Board Rule 41.37, Appellants present this brief and enclose herewith a check for the fee of $500.00 required under 37 C.F.R. § 1.17(c).

This Appeal responds to the final rejection of claims 1-2, 5-11, 13-22, 25-29, 31-33, 37, 38, 41-47 and 49-53 (all currently pending claims) in the Office Action dated May 5, 2005.

If any additional fees are required or if the enclosed payment is insufficient, Appellants request that the required fees be charged to Deposit Account No. 06-0916.

10/06/2005 JADD01 00000009 09895077
01 FC:1402 500.00 OP

TABLE OF CONTENTS

G. The rejection of dependent claims 8, 28 and 44 under 35 U.S.C. 102(b) must be reversed because Mitchell et al. fails to teach, *for each new class instance and corresponding association class instance,* determining all instances of the association class that reference the new class instance, and creating a pointer in the new second level wrapper table for each instance of the association class. 39

H. The rejection of claims 9 and 45 under 35 U.S.C. 102(b) must be reversed because Mitchell et al. fails to teach, *for each class instance associated with N association class classes,* creating a first level wrapper table including N pointers to N second level wrapper tables associated with the association classes..... 40

I. The rejection of claims 10 and 46 under 35 U.S.C. 102(b) must be reversed because Mitchell et al. fails to teach, *for each class instance,* creating a first level wrapper table including a second pointer to a second level wrapper table associated with a second association class, and creating X pointers in the second level wrapper table that each reference an individual instance of the second association class..... 41

J. The rejection of claims 11, 29 and 47 under 35 U.S.C. 102(b) must be reversed because Mitchell et al. fails to teach receiving an association traversal request for a class instance, and performing an association traversal process including (1) accessing a first table

I. REAL PARTY IN INTEREST

The real party in interest is Sun Microsystems, Inc., the assignee of the entire right, title and interest in the present Application.

II. RELATED APPEALS AND INTERFERENCES

There are currently no other appeals or interferences, of which Appellants, Appellants' legal representative, or Assignee are aware, that will directly affect or be directly affected by or have a bearing on the Board's decision in the pending appeal.

III. STATUS OF CLAIMS

Claims 1-2, 5-11, 13-22, 25-29, 31-33, 37, 38, 41-47 and 49-53 remain pending. Claims 3, 4, 12, 23, 24, 30, 34-36, 39, 40, 48 and 55-56 have been cancelled. The Final Rejection of claims 1-2, 5-11, 13-22, 25-29, 31-33, 37, 38, 41-47 and 49-53 is appealed.

IV. STATUS OF AMENDMENTS

An Amendment After Final was filed July 5, 2005. In the Advisory Action dated July 27, 2005, the Examiner indicated that this Amendment will be entered for purposes of appeal.

V. SUMMARY OF CLAIMED SUBJECT MATTER

In accordance with Rule 41.37(c)(1)(v), Appellants present a concise explanation of the subject matter defined in each of the independent claims involved in the appeal, referring to the specification by page and line number and the drawings by reference characters. For each independent claim involved in the appeal and for each dependent claim argued separately, Appellant also identifies every means-plus-function element and sets forth the structure, material or acts corresponding to the claimed function with reference to the specification by page and line number and to the drawing by reference character. Unless otherwise indicated, the structure for each identified means-plus-function element is identified in brackets "[ ]." Appellants note that the structures or acts identified below are representative or are one of several combinations of structures or acts corresponding to the recited means and in no way limit the scope of these claims as to 35 U.S.C. § 112, sixth paragraph, which specifically provides for coverage of equivalents of these structures or acts.

A. Independent Claim 1

Independent claim 1 is directed to a computer-implemented method for determining a relationship between objects (see, e.g., FIG. 4, elements 420 and 450; p. 10, ¶ 043, ll. 1-2) related to a common information model (see, e.g., FIGS. 1 and 2; p. 2, ¶ 002, ll. 1-2). The objects include at least a first and second instance (see, e.g., FIG. 6A, elements 610 and 622; p. 16, ll. 1-4) and an association that represents an instance of an association class (see, e.g., FIG. 6A, element 612; p. 16, ll. 4-8). The method comprises creating, for the first instance, a reverse link that defines a

relationship between the first instance and the association (see, e.g., p. 11, ¶ 044, ll. 1-5). The instance is associated with a first wrapper defining the reverse link (see, e.g., p. 16, ll. 8-10). The method further comprises determining a relationship between the first and second instances based on the reverse link (see, e.g., p. 5, ¶ 014, ll. 1-5). Creating the reverse link includes defining a pointer in a first table (e.g., FIG. 6B, element 632; p. 16, ¶ 057, ll. 1-2) of the first wrapper that references a second table (see, e.g., FIG. 6B, element 644; p. 16, ¶ 057, ll. 6-8) and defining a pointer in the second table that references the instance of the association class (see, e.g., p. 17, ¶ 059, ll. 2-6).

B. Independent Claim 6

Independent claim 6 is directed to a method for maintaining reverse links (see, e.g., FIG. 4, element 440; p. 11, ¶ 044, ll. 1-5) in a object-oriented environment including class instances (see, e.g., FIG. 6A, elements 610 and 622-630; p. 16, ll. 1-4) and associations (see, e.g., FIG. 6A, elements 612-620; p. 16, ll. 4-8). The method comprises, for each class instance associated with N instances of an association class that each reference the class instance (e.g., AC1: see, e.g., FIG. 6A, elements 612-616; p. 16, ll. 6-8), wherein N represents an integer value greater than or equal to one, (i) creating a first level wrapper table (e.g., FIG. 6B, element 632; p. 16, ¶ 057, ll. 6-8) including a pointer to a second level wrapper table (e.g., FIG. 6B, element 644; p. 16, ¶ 057, ll. 6-8) associated with the association class, and (ii) creating N pointers in the second level wrapper table that each reference an individual instance of the association class (see, e.g., p. 17, ¶ 059, ll. 2-4).

C. Independent Claim 9

Independent claim 9 is directed to a method for maintaining reverse links (see, e.g., FIG. 4, element 440; p. 11, ¶ 044, ll. 1-5) in an object-oriented environment including instances (see, e.g., FIG. 6A, elements 610 and 622-630; p. 16, ll. 1-4), association classes (e.g., A-1 to A-3; see, e.g., p. 15, ¶ 056, ll. 1-3) and association class instances (see, e.g., FIG. 6A, elements 612-620; p. 16, ll. 4-8). The method comprises, for each class instance associated with N association class classes that each include at least one association class instance that references the class instance (e.g., AC1: see, e.g., FIG. 6A, elements 612-616; p. 16, ll. 6-8), wherein N represents an integer value greater than or equal to one, creating a first level wrapper table (e.g., FIG. 6B, element 632; p. 16, ¶ 057, ll. 1-2) including N pointers to N second level wrapper tables associated with the association classes (e.g., FIG. 6B, element 644; p. 16, ¶ 057, ll. 6-8).

D. Independent Claim 10

Independent claim 10 is directed to a method for maintaining reverse links (see, e.g., FIG. 4, element 440; p. 11, ¶ 044, ll. 1-5) in a object-oriented environment including class instances (see, e.g., FIG. 6A, elements 610 and 622-630; p. 16, ll. 1-4) and associations (see, e.g., FIG. 6A, elements 612-620; p. 16, ll. 4-8). The method comprises, for each class instance associated with N instances of a first association class (e.g., AC1: see, e.g., FIG. 6A, elements 612-616; p. 16, ll. 6-8) that each reference the class instance, and X instances of a second association class (e.g., AC2: see, e.g., FIG. 6A, element 618; p. 16, ll. 6-8) that each reference the class instance,

wherein N and X represent integer values greater than or equal to one: (i) creating a first level wrapper table (see, e.g., FIG. 6B, element 632; p. 16, ¶ 057, ll. 1-8) including a first pointer to a second level wrapper table (see, e.g., FIG. 6B, element 644; p. 16, ¶ 057, ll. 6-8) associated with the first association class, and a second pointer to a second level wrapper table (see, e.g., FIG. 6B, element 646; p. 16, ¶ 057, ll. 6-8) associated with the second association class; (ii) creating N pointers, in the second level wrapper table associated with the first association class, that each reference an individual instance of the first association class (see, e.g., p. 17, ¶ 059, ll. 1-4); and (iii) creating X pointers, in the second table level wrapper table associated with the second association class, that each reference an individual instance of the second association class (see, e.g., p. 17, ¶ 059, ll. 1-4).

E. Independent Claim 11

Independent claim 11 is directed to a method for performing association traversals in an object-oriented environment including a plurality of class instances (see, e.g., FIG. 6A, elements 610 and 622-630; p. 16, ll. 1-4) and association instances (see, e.g., FIG. 6A, elements 612-620; p. 16, ll. 4-8). The method comprises receiving an association traversal request for a class instance (see, e.g., FIG. 8, element 810; p. 18, ¶ 064, ll. 3-4); and performing an association traversal process based on pointer information reflecting a relationship between the class instance and all association instances that reference the class instance (see, e.g., p. 18, ¶ 064, ll. 1-3). The association traversal process includes: accessing a first table (see, e.g., FIG. 8, element 820; p. 18, ¶ 064, ll. 4-6) including a pointer to a second table (see, e.g., FIG.

8, element 840; p. 19, ll. 2-4); and accessing the second table, using the pointer (see, e.g., FIG. 8, element 850; p. 19, ll. 4-6), to obtain pointers to each association instance that references the class instance (see, e.g., FIG. 8, element 855; p. 19, ¶ 065, ll. 1-3).

F. Independent Claim 14

Independent claim 14 is directed to a method for performing association traversals in an object-oriented environment including a plurality of class instances (see, e.g., FIG. 6A, elements 610 and 622-630; p. 16, ll. 1-4) and association instances (see, e.g., FIG. 6A, elements 612-620; p. 16, ll. 4-8). The method comprises: receiving an association traversal request for a class instance (see, e.g., FIG. 8, element 810; p. 18, ¶ 064, ll. 3-4); obtaining pointers to each association instance that references the class instance from a wrapper table corresponding to the class instance (see, e.g., FIG. 8, element 840; p. 19, ll. 2-4); and collecting, from the wrapper table, references to other class instances that are referenced by the association instance (see, e.g., FIG. 8, element 855; p. 19, ¶ 065, ll. 1-3).

G. Independent Claim 15

Independent claim 15 is directed to a method for performing association traversals in an object-oriented environment including a plurality of class instances (see, e.g., FIG. 6A, elements 610 and 622-630; p. 16, ll. 1-4) and association instances (see, e.g., FIG. 6A, elements 612-620; p. 16, ll. 4-8). The method comprises: receiving an association traversal request for a class instance (see, e.g., FIG. 8, element 810; p. 18, ¶ 064, ll. 3-4); and obtaining pointers to each association instance that references the

class instance from a wrapper table corresponding to the class instance (see, e.g., FIG. 8, element 840; p. 19, ll. 2-4).

H. Independent Claim 16

Independent claim 16 is directed to a method for performing association traversals performed by the client in a system comprising a client (see, e.g., FIG. 3, element 320; p. 9, ¶ 037, ll. 1-2) and a server (see, e.g., FIG. 3, element 330; p. 8, ¶ 036, ll. 1-3). The method comprises: generating a request for relationship information associated with a selected class instance (see, e.g., p. 18, ¶ 064, ll. 3-4); and receiving a response including information reflecting a relationship between the selected class instance and other class instances that are referenced by the same association class instances (see, e.g., p. 19, ¶ 065, ll. 8-12). The response is generated using pointers defined in a table associated with the selected class instance that reference the common association class instances (see, e.g., FIG. 8, element 840; p. 19, ll. 2-4).

I. Independent Claim 17

Independent claim 17 is directed to a method for performing association traversals performed by the server in a system comprising a client (see, e.g., FIG. 3, element 320; p. 9, ¶ 037, ll. 1-2) and a server (see, e.g., FIG. 3, element 330; p. 8, ¶ 036, ll. 1-3). The method comprises: receiving a request for relationship information associated with a selected class instance (see, e.g., FIG. 8, element 810; p. 18, ¶ 064, ll. 3-4); and generating a response including information reflecting a relationship between the selected class instance and other class instances that are referenced by the same association class instances (see, e.g., FIG. 8, element 880; p. 19, ¶ 065,

ll. 8-10), based on pointers defined in a table associated with the selected class instance that reference the common association class instances (see, e.g., FIG. 8, element 840; p. 19, ll. 2-4).

J. Independent Claim 18

Independent claim 18 is directed to a system for performing association traversals. The system comprises a client (see, e.g., FIG. 3, element 320; p. 9, ¶ 037, ll. 1-2) for generating a request for information reflecting the relationship between selected objects (see, e.g., FIG. 4, elements 420 and 450) defined in a repository (see, e.g., FIG. 3, element 350; p. 10, ¶ 040, ll. 1-3). The system also comprises a server (see, e.g., FIG. 3, element 330; p. 8, ¶ 036, ll. 1-3). The server includes the repository for storing objects (see, e.g., p. 10, ¶ 043, ll. 1-2), wherein a set of the objects are associated with object wrappers (see, e.g., FIG. 4, element 410; p. 11, ll. 11-12), and an object manager (see, e.g., FIG. 3, element 340; p. 9, ¶ 039, ll. 1-2) for processing the request by using an object wrapper associated with one object of the selected objects (see, e.g., p. 20, ¶ 068, ll. 3-6).

K. Independent Claim 21 and Dependent Claim 25

Independent claim 21 is directed to a system for traversing associations in a common information model (see, e.g., FIGS 1 and 2; p. 2, ¶002, ll. 1-2) implemented environment. The model comprises at least a first and second instance (see, e.g., FIG. 6A, elements 610 and 622; p. 16, ll. 1-4) and an association that represents an instance of an association class (see, e.g., FIG. 6A, element 612, p. 16, ll. 4-8). The system comprises means [see, e.g., object manager 340; p. 8, ¶ 034, ll. 7-9] for creating, for the

first instance, a reverse link that defines a relationship between the first instance and the association (see, e.g., p. 11, ¶044, ll. 1-5), wherein the instance is associated with a first wrapper defining the reverse link (see, e.g., p. 16, ll. 8-10). The means for creating the reverse link includes means [see, e.g., FIG. 6B, entries in column 633; p. 16, ¶ 057, ll. 4-6] for defining a pointer in a first table (e.g., FIG. 6B, element 632; p. 16, ¶057, ll. 1-2) of the first wrapper that references a second table (see, e.g., FIG. 6B, element 644; p. 16, ¶057, ll. 6-8); means [see, e.g., FIG. 6B, entries in table 644; p. 17, ¶ 059, ll. 2-4] for defining a pointer in the second table that references the instance of the association class (see, e.g., p. 17, ¶059, ll. 2-6); and means for determining a relationship between the first and second instances based on the reverse link [see, e.g., object manager 340; p. 5, ¶014, ll. 1-5].

Dependent claim 25 is directed to the system of claim 21, wherein the means for determining a relationship includes means for collecting a reference reflecting a relationship between the association and the second instance based on the pointer in the second table [see, e.g., object manager 340; FIG. 8, element 855; p. 19, ¶ 065, ll. 1-3].

L. Independent Claim 26 and Dependent Claims 27 and 28

Independent claim 26 is directed to a system for maintaining reverse links (see, e.g., FIG. 4, element 440; p. 11, ¶044, ll. 1-5) in an object-oriented environment including class instances (see, e.g., FIG. 6A elements 610 and 622-630; p. 16, ll. 1-4) and associations (see, e.g., FIG. 6A, elements 612-620; p. 16 ll. 4-8). The system comprises means [see, e.g., object manager 340; p. 14, ll. 3-5] for creating, for each

class instance associated with N instances of an association class that each reference the class instance (e.g., AC1; see, e.g., FIG. 6A, elements 612-616; p. 16, ll. 6-8), wherein N represents an integer value greater than or equal to one, a first level wrapper table (e.g., FIG 6B, element 632; p. 16, ¶057, ll. 6-8) including a pointer to a second level wrapper table (e.g., FIG. 6B, element 644; p. 16, ¶057, ll. 6-8) associated with the association class. The system also comprises means [see, e.g., object manager 340; p. 14, ll. 3-5] for creating N pointers in the second level wrapper table that each point to an individual instance of the association class (see, e.g., p. 17, ¶059, ll. 2-4).

Dependent claim 27 is directed to the system of claim 26, further comprising: means [see, e.g., object manager 340; p. 14, ll. 3-5] for creating, for each new class instance (e.g., FIG. 6D, element 680; p. 17, ¶ 061, ll. 1-2) and corresponding new association class instance (e.g., FIG. 6D, element 670; p. 17, ¶ 061, ll. 1-2) that reference the new class instance that is created, a new first level wrapper table (e.g., FIG. 6B, element 632; p. 16, ¶ 057, ll. 1-2) including a pointer to a new second level wrapper table associated with the association class (see, e.g., FIG. 6B, element 644; p. 16, ¶ 057, ll. 6-8); and means [see, e.g., object manager 340; p. 14, ll. 3-5] for creating a pointer in the new second level wrapper table that references the new instance of the association class (see, e.g., p. 17, ¶ 059, ll. 1-4).

Dependent claim 28 is directed to the system of claim 26, further comprising: means [see, e.g., FIG. 8, steps 820-870; p. 18, ¶ 064, l. 4-p. 19, ¶ 065, l. 8] for determining, for each new class instance and corresponding new association class instance that is created, all instances of the association class that reference the new class instance; means [see, e.g., object manager 340; p. 14, ll. 3-5] for creating a new

first level wrapper table (e.g., FIG. 6B, element 632; p. 16, ¶ 057, ll. 1-2) including a pointer to a new second level wrapper table associated with the association class (see, e.g., FIG. 6B, element 644; p. 16, ¶ 057, ll. 6-8); and means [see, e.g., object manager 340; p. 14, ll. 3-5] for creating a pointer in the new second level wrapper table for each instance of the association class determined by the means for determining (see, e.g., p. 17, ¶ 059, ll. 1-4).

M. Independent Claim 29 and Dependent Claim 31

Independent claim 29 is directed to a system for performing association traversals in an object-oriented environment including a plurality of class instances (see, e.g., FIG. 6A, elements 610 and 622-630; p. 16, ll. 1-4) and association instances (see, e.g., FIG. 6A, elements 612-620; p. 16, ll. 4-8). The system comprises: means [see, e.g., object manager 340; FIG. 8, step 810; p. 18, ¶064, ll. 3-4] for receiving an association traversal request for a class instance; and means for performing an association traversal process based on pointer information reflecting a relationship between the class instance and all association instances that reference the class instance [see, e.g., object manager 340; p. 18, ¶064, ll. 1-3]. The means for performing an association traversal process includes: means [see, e.g., object manager 340; FIG. 8, step 820; p. 18, ¶064, ll. 4-6] for accessing a first table including a pointer to a second table (see, e.g., FIG. 8, element 840; p. 19, ll. 2-4); and means [see, e.g., object manager 340; FIG. 8, element 850; p. 19, ll. 4-6] for accessing the second table, using the pointer, to obtain pointers to each association instance that references the class instance (see, e.g., FIG. 8, element 855; p. 19, ¶065, ll. 1-3).

Dependent claim 31 is directed to the system of claim 29, wherein the means for performing an association traversal process further includes: means [see, e.g., object manager 340; FIG. 8, element 855; p. 19, ¶ 065, ll. 1-3] for collecting, for each association instance pointed to by the second table, a reference to another class instance that is referenced by the association instance.

N. Independent Claim 32

Independent claim 32 is directed to a system for performing association traversals in an object-oriented environment including a plurality of class instances (see, e.g., FIG. 6A, elements 610 and 622-630; p. 16, ll. 1-4) and association instances (see, e.g., FIG. 6A, elements 612-620; p. 16, ll. 4-8). The system comprises: means [see, e.g., object manager 340; FIG. 8, step 810; p. 18, ¶064, ll. 3-4] for receiving an association traversal request for a class instance; means [see, e.g., object manager 340; FIG. 8, step 840; p. 19, ll. 2-4] for obtaining pointers to each association instance that references the class instance from a wrapper table corresponding to the class instance; and means [see, e.g., object manager 340; FIG. 8, element 855; p. 19, ¶065, ll. 1-3] for collecting, from the wrapper table, references to other class instances that are referenced by the association instance.

O. Independent Claim 33

Independent claim 33 is directed to a system for performing association traversals in an object-oriented environment including a plurality of class instances (see, e.g., FIG. 6A, elements 610 and 622-630; p. 16, ll. 1-4) and association instances (see, e.g., FIG. 6A, elements 612-620; p. 16, ll. 4-8). The system comprises: means [see,

e.g., object manager 340; FIG. 8, step 810; p. 18, ¶064, ll. 3-4] for receiving an association traversal request for a class instance; and means [see, e.g., object manager 340; FIG. 8, step 840; p. 19, ll. 2-4] for obtaining pointers to each association instance that references the class instance from a wrapper table corresponding to the class instance.

P. Independent Claim 37

Independent claim 37 is directed to a computer-readable medium including instructions for performing a method, when executed by a processor, for determining a relationship between objects (see, e.g., FIG. 4, elements 420 and 450; p. 10, ¶ 043, ll. 1-2) related to a common information model (see, e.g., FIGS. 1 and 2; p. 2, ¶002, ll. 1-2). The objects include at least a first and second instance (see, e.g., FIG. 6A, elements 610 and 622; p. 16, ll. 1-4) and an association that represents an instance of an association class (see, e.g., FIG. 6A, element 612; p. 16, ll. 4-8). The method comprises: creating, for the first instance, a reverse link that defines a relationship between the first instance and the association (see, e.g., p. 11, ¶044, ll. 1-5), wherein the instance is associated with a first wrapper defining the reverse link (see, e.g., p. 16, ll. 8-10) and determining a relationship between the first and second instances based on the reverse link (see, e.g., p. 5, ¶014, ll. 1-5). Creating the reverse link includes: defining a pointer in a first table (e.g., FIG 6B, element 632; p. 16, ¶057, ll. 1-2) of the first wrapper that references a second table (see, e.g., FIG. 6B, element 644; p. 16, ¶057, ll. 6-8); and defining a pointer in the second table that references the instance of the association class (see, e.g., p. 17, ¶059, ll. 2-6).

Q. Independent Claim 42

Independent claim 42 is directed to a computer-readable medium including instructions for performing a method, when executed by a processor, for maintaining reverse links in a object-oriented environment including class instances (see, e.g., FIG. 6A, elements 612 and 622-630; p. 16, ll. 1-4) and associations (see, e.g., FIG. 6A elements 610 and 622-630; p. 16, ll. 4-8). The method comprises, for each class instance associated with N instances of an association class that each reference the class instance (e.g., AC1: see, e.g., FIG. 6A, elements 612-616; p. 16, ll. 6-8), wherein N represents an integer value greater than or equal to one, (i) creating a first level wrapper table (e.g., FIG. 6B, element 632; p. 16, ¶057, ll. 6-8) including a pointer to a second level wrapper table (e.g., FIG. 6B, element 644; p. 16, ¶057, ll. 6-8) associated with the association class; and (ii) creating N pointers in the second level wrapper table that each reference an individual instance of the association class (see, e.g., p. 17, ¶059, ll. 2-4).

R. Independent Claim 45

Independent claim 45 is directed to a computer-readable medium including instructions for performing a method, when executed by a processor, for maintaining reverse links (see, e.g., FIG. 4, element 440; p. 11, ¶044, ll. 1-5) in a object-oriented environment including class instances (see, e.g., FIG. 6A elements 610 and 622-630; p. 16, ll. 1-4) and associations (e.g., A-1 to A-3; see, e.g., p. 15, ¶056, ll. 1-3). The method comprises, for each class instance associated with N association classes that each includes include at least one association class instance that references the class

instance (e.g., AC1: see, e.g., FIG. 6A, elements 612-616; p. 16, ll. 6-8), wherein N represents an integer value greater than or equal to one: creating a first level wrapper table (e.g., FIG. 6B, element 632; p. 16, ¶057, ll. 1-2) including N pointers to N second level wrapper tables associated with the association classes (e.g., FIG. 6B, element 644; p. 16, ¶057, ll. 6-8).

S. Independent Claim 46

Independent claim 46 is directed to a computer-readable medium including instructions for performing a method, when executed by a processor, for maintaining reverse links (see, e.g., FIG. 4, element 440; p. 11, ¶044, ll. 1-5) in a object-oriented environment including class instances (see, e.g., FIG. 6A, elements 610 and 622-630; p. 16, ll. 1-4) and associations (see, e.g., FIG. 6A, elements 612-620; p. 16 ll. 4-8). The method comprises, for each class instance associated with N instances of a first association class (e.g., AC1: see, e.g., FIG. 6A, elements 612-616; p. 16, ll. 6-8) that each reference the class instance, and X instances of a second association class (e.g., AC2: see, e.g., FIG. 6A, element 618; p. 16, ll. 6-8) that each reference the class instance, wherein N and X represent integer values greater than or equal to one, (i) creating a first level wrapper table (see, e.g., FIG. 6B, element 632; p. 16, ¶057, ll. 1-8) including a first pointer to a second level wrapper table (see, e.g., FIG. 6B, element 644, p. 16, ¶057, ll. 6-8) associated with the first association class, and a second pointer to a second level wrapper table (see, e.g., FIG. 6B, element 646; p. 16, ¶057, ll. 6-8) associated with the second association class; (ii) creating N pointers, in the second level wrapper table associated with the first association class , that each

reference an individual instance of the first association class (see, e.g., p. 17, ¶059, ll. 1-4); and (iii) creating X pointers, in the second table level wrapper table associated with the second association class, that each reference an individual instance of the second association class (see, e.g., p. 17, ¶059, ll. 1-4).

T. Independent Claim 47

Independent claim 47 is directed to a computer-readable medium including instructions for performing a method, when executed by a processor, for performing association traversals in an object-oriented environment including a plurality of class instances (see, e.g., FIG. 6A, elements 610 and 622-630; p. 16, ll. 1-4) and association instances (see, e.g., FIG. 6A, elements 622-620; p. 16, ll. 4-8). The method comprises: receiving an association traversal request for a class instance (see, e.g., FIG. 8, element 810; p. 18, ¶064, ll. 3-4); and performing an association traversal process based on pointer information reflecting a relationship between the class instance and all association instances that reference the class instance (see, e.g., p. 18, ¶064, ll. 1-3). The association traversal process includes: accessing a first table (see, e.g., FIG. 8, element 820; p. 18, ¶064, ll. 4-6) including a pointer to a second table (see, e.g., FIG. 8, element 840, p. 19, ll. 2-4); and accessing the second table (see, e.g., FIG. 8, element 840; p. 19, ll. 4-6), using the pointer (see, e.g., FIG. 8, element 850; p. 19, ll. 4-6), to obtain pointers to each association instance that references the class instance (see, e.g., FIG. 8, element 855; p. 19, ¶065 ll. 1-3).

U. Independent Claim 50

Independent claim 50 is directed to a computer-readable medium including instructions for performing a method, when executed by a processor, for performing association traversals in an object-oriented environment including a plurality of class instances (see, e.g., FIG. 6A, elements 610 and 622-630; p. 16, ll. 1-4) and association instances (see e.g., FIG. 6A, elements 612-620; p. 16, ll. 4-8). The method comprises: receiving an association traversal request for a class instance (see, e.g., FIG. 8, element 810; p. 18, ¶064, ll. 3-4); obtaining pointers to each association instance that references the class instance from a wrapper table corresponding to the class instance (see, e.g., FIG. 8, element 840, p. 19, ll. 2-4); and collecting, from the wrapper table, references to other class instances that are referenced by the association instance (see, e.g., FIG. 8, element 855; p. 19, ¶065, ll. 1-3).

V. Independent Claim 51

Independent claim 51 is directed to a computer-readable medium including instructions for performing a method, when executed by a processor, for performing association traversals in an object-oriented environment including a plurality of class instances (see, e.g., FIG. 6A, elements 610 and 622-630; p. 16, ll. 1-4) and association instances (see, e.g., FIG. 6A, elements 612-620; p. 16, ll. 4-8). The method comprises: receiving an association traversal request for a class instance (see, e.g., FIG. 8, element 810; p. 18, ¶064, ll. 3-4); and obtaining pointers to each association instance that references the class instance from a wrapper table corresponding to the class instance (see, e.g., FIG. 8, element 840; p. 19, ll. 2-4).

W. Independent Claim 52

Independent claim 52 is directed to a computer-readable medium including instructions for performing a method, when executed by a processor, for performing association traversals performed by the client in a system comprising a client (see, e.g., FIG. 3, element 320; p. 9, ¶037, ll. 1-2) and a server (see, e.g., FIG. 3, element 330; p. 8, ¶036, ll. 1-3). The method comprises: generating a request for relationship information associated with a selected class instance (see, e.g., p. 18, ¶064, ll. 3-4); receiving a response including information reflecting a relationship between the selected class instance and other class instances that are referenced by the same association class instances (see, e.g., p. 19, ¶064, ll. 8-12), wherein the response is generated using pointers defined in a table associated with the selected class instance that reference the common association class instances (see, e.g., FIG. 8, element 840; p. 19, ll. 2-4).

X. Independent Claim 53

Independent claim 53 is directed to a computer-readable medium including instructions for performing a method, when executed by a processor, for performing association traversals performed by the server a system comprising a client (see, e.g., FIG. 3, element 320; p. 9, ¶037, ll. 1-2) and a server (see, e.g., FIG. 3, element 330; p. 8, ¶036, ll. 1-3). The method comprises: receiving a request for relationship information associated with a selected class instance (see, e.g., FIG. 8, element 810; p. 18, ¶064, ll. 3-4); generating a response including information reflecting a relationship between the selected class instance and other class instances that are referenced by

the same association class instances (see, e.g., FIG. 8 element 880; p. 19, ¶065, ll. 8-10), based on pointers defined in a table associated with the selected class instance that reference the common association class instances (see, e.g., FIG. 8, element 840; p. 19, ll. 2-4).

VI. GROUNDS OF REJECTION TO BE REVIEWED

Claims 1, 2, 4-11, 13-22, 24-29, 31-33, 37, 38, 40-47 and 49-53 (i.e., all claims currently pending) stand rejected under 35 U.S.C. § 102(b) as being unpatentable over Mitchell et al. (U.S. Patent No. 5,872,973).

VII. ARGUMENT

A. The rejection of claims 1, 21 and 37 under 35 U.S.C. 102(b) must be reversed because Mitchell et al. fails to teach (1) creating, for a first instance, a reverse link that defines a relationship between the first instance and an association, wherein the instance is associated with a first wrapper defining the reverse link, and (2) determining a relationship between the first instance and a second instance based on the reverse link.

In the rejection of claims 1, 21 and 37, the Examiner asserts that Mitchell et al. teaches:

> a first [Patron Object 102] and second instance [Patron Object 106] and an association [103 ('mapper' or 'mapping object')]
>
> creating, for the first instance [e.g., 102], a reverse link [Probe] that defines a relationship between the first instance and the association [see Figs. 1-2 & Column 8, lines 10-32] wherein the instance [e.g. 102 or 106] is associated with a first wrapper [EosConnection (See Column 16, lines 10-26)] defining the reverse link; and
>
> determining a relationship between the first and second instances based on the reverse link [see Appendix C and Column 14].

Final Office Action, p. 3, ll. 2-10 (square-bracketed material in original); see also id., p. 7, ll. 1-3 and 7-9. Appellants respectfully disagree with the Examiner's characterization of Mitchell et al.

The "probes" of Mitchell et al. (which the Examiner attempts to read on the "reverse link" recited in claims 1, 21 and 37) "are callback functions that are invoked when data (typically in an object field) change." Mitchell et al., col. 30, ll. 52-57. For example, mapper 103 of Mitchell et al. may synchronize the value in a field 101 in a first patron object 102 with the value in another field 105 in a second patron object 106 by

setting corresponding "probes" on fields 101 and 105. See id., col. 8, ll. 49-56; and col. 14, ll. 5-8. When the value in field 105 is changed, its probe "fires" and mapper 103 "simply gets the values from the source side parameter [e.g., field 105] (that changed) and sets the value on the destination side [field 101] to this new value." Id., col. 14, ll. 34-38.

Contrary to the Examiner's assertions, Mitchell et al. does not "determine a relationship" between the first and second patron objects (which the Examiner attempts to read on the recited instances) based on the probes. In fact, Mitchell et al. specifies that "[t]he mapper [103] is never referenced or visible to ... the patron objects" (id. *col.* 9, ll. 15-16), and "the [patron] object has no knowledge of which objects (if any) have placed [the] probes" (id., col. 31, ll. 36-37). Moreover, while the EosConnection described by Mitchell et al. is described as a "wrapper" (id., col. 16, l. 11), it is not a wrapper "defining" *the probe*. Instead, it is "a simple wrapper around *the RPC* [Remote Procedure Call] *calls used in the network connection*." Id., col. 16, ll. 11-12 (emphasis added).

For at least these reasons, Mitchell et al. fails to support the Examiner's rejection of claims 1, 21 and 37 under 35 U.S.C. § 102(b). Accordingly, Appellants respectfully request that the rejection of these claims be reversed and the claims allowed.

B. The rejection of claims 1, 21 and 37 under 35 U.S.C. 102(b) must be reversed because Mitchell et al. fails to teach creating a reverse link by defining a pointer in a first table . . . that references a second table and defining a pointer in a second table that references the instance of the association class.

In the rejection of claims 1, 21 and 37, the Examiner further asserts that Mitchell et al. discloses:

> defining a pointer [probe by field or path, termed 'connection descriptor'] in a first table that references a second table [see Column 22, line 60 – Column 23, line 48]; and defining a pointer ['semantic link'] in the second table ['the table containing the connection descriptor' (see Column 21, line 34 – Column 23, line 48)] that references that *instance* [EosMapFieldtoField (e.g., 103)] of the association class [EosMapElement] as claimed.

Final Office Action, p. 3, l. 21, through p. 4, l. 4 (square-bracketed material in original) (emphasis added); see also id., p. 7, ll. 1-3 and 7-9. Appellants respectfully disagree with the above characterization of Mitchell et al. for the following reasons.

First, Appellants can find no teaching in Mitchell et al. (and the Examiner has pointed to none) indicating that the probe (which the Examiner attempts to read on the recited pointer in the first table) references the table with the connection descriptor (which the Examiner attempts to read on the recited second table). Rather, as shown in FIG. 1, the probes point to mapper 103 (which the Examiner attempts to read as the recited instance of the association class). See Mitchell et al., col. 14, ll. 44-46.

Second, the portion of Mitchell et al. cited by the Examiner refers to "the table containing the connection *descriptor*." Mitchell et al., col. 22, ll. 25-26. Thus, the table does not "contain" the semantic link, *per se*, as asserted by the Examiner. Rather, it contains "the semantic link *description*." Id. at col. 22, ll. 27-28 (emphasis added).

Contrary to the Examiner's arguments (see Final Office Action, p. 9, ll. 4-8), this table is used to *initialize* the semantic links:

> The edit function interprets the information in *the table* to first find the class for which the connection is stored, then find the named connection by looking it up in the table of connections in each class, then reading the specification for the connection out of the table and creating and *initializing* the proper data structures to set up the individual semantic links as described in the previously shown link.

Mitchell et al., col. 21, ll. 41-47 (emphasis added). Consequently, the "semantic link description" (which the Examiner attempts to read on the recited pointer in second table) does not reference an *instance* of the EosMapElement class (which the Examiner attempts to read on the recited association class). Instead, the semantic link description is a generic description of connections between classes of objects, that is used to initialize the semantic links. Id.

For at least these additional reasons, Mitchell et al. fails to support the Examiner's rejection of claims 1, 21 and 37 under 35 U.S.C. § 102(b). Accordingly, Appellants respectfully request that the rejection of these claims be reversed and the claims allowed.

C. The rejection of dependent claims 2, 22 and 38 under 35 U.S.C. 102(b) must be reversed for at least the same reasons given above with respect to claims 1, 21 and 37.

Claims 2, 22 and 38 depend from claims 1, 21 and 37, respectively. As explained in Sections A and B above, the Examiner's rejection of claims 1, 21 and 37 under 35 U.S.C. § 102(b) is not supported by Mitchell et al. Therefore, the Examiner's rejection of claims 2, 5, 38 and 41 under 35 U.S.C. § 102(b) is likewise not supported by

Mitchell et al. for at least the same reasons given above. Accordingly, the rejection of these claims must be reversed and the claims allowed.

D. The rejection of dependent claims 5, 25 and 41 under 35 U.S.C. 102(b) must be reversed because Mitchell et al. fails to teach collecting a reference reflecting a relationship between the association and the second instance based on a pointer in a second table.

Claims 5, 25 and 41 depend from claims 1, 21 and 37, respectively. As explained in Sections A and B above, the Examiner's rejection of claims 1, 21 and 37 under 35 U.S.C. § 102(b) is not supported by Mitchell et al. Therefore, the Examiner's rejection of claims 5, 25 and 41 under 35 U.S.C. § 102(b) is likewise not supported by Mitchell et al. for at least the same reasons given above. Accordingly, the rejection of these claims must be reversed and the claims allowed.

Moreover, in the rejection of claims 5, 25 and 41, the Examiner asserts that Mitchell et al. teaches "determining a relationship {between the first and second instances} includ[ing]: collecting a reference [fRightSide] reflecting a relationship between the association [103] and the second instance [106] based on the pointer in the second table." See Final Office Action, p. 4, ll. 6-9 (square-bracketed material in original); see also id., p. 7, ll. 1-3 and 7-9. Appellants respectfully disagree with the Examiner's interpretation of Mitchell et al.

Contrary to the Examiner's assertions, the member fRightSide described by Mitchell et al. does not reflect a relationship between the mapper 103 (which the Examiner attempts to read on the recited association) and an *instance* of the patron

object 106. Instead, it represents the field on which a probe is to be set when the semantic link is *initialized*:

> The object view mapper object now traverses the table containing the connection descriptor. There is an entry for each semantic link. For each semantic link description, the individual mappers are created by name through the generic factory method and initialized using property information stored in the table for each semantic link. This includes the creation and initialization of the EosTypeElement members (fLeftSide and fRightSide) and their respective path objects. These like the mappers are created by type name according to the information stored in the descriptor information.

Mitchell et al., col. 22, ll. 25-34; see also id., col. 22, l. 61-col. 23, l. 7. Therefore, Mitchell et al. does not teach determining a relationship between first and second *instances*, including collecting a reference reflecting a relationship between an association (that represents an *instance* of the association class) and the second *instance* based on a pointer in a second table.

For at least these additional reasons, the Examiner's rejection of claims 5, 25 and 41 is not supported by Mitchell et al. Accordingly, the rejection of these claims under 35 U.S.C. § 102(b) must be reversed and the claims allowed.

E. The rejection of claims 6, 26 and 42 under 35 U.S.C. 102(b) must be reversed because Mitchell et al. fails to teach, *for each class instance,* creating a first level wrapper table including a pointer to a second level wrapper table and creating pointers in the second level wrapper table that each reference an individual instance of the association class.

In the rejection of claims 6, 26 and 42, the Examiner asserts that Mitchell et al. teaches:

> for *each* class *instance* . . .
>
> (i) creating a first level wrapper table {citing Mitchell et al., col. 22, l. 60, through col. 23, l. 48} including a pointer [probe] to a second level wrapper table ['the table containing the connection descriptor' (See Column 21, line 34 – Column 23, line 48)] associated with the association class; and
>
> (ii) creating N [one entry for each mapper object present in the table] pointers ['semantic links'] in the second level wrapper table [see above & Claim 4] that each references an *individual instance* [EosMapFieldToField (e.g. 103)] of the association class as claimed.

Final Office Action, p. 4, l. 21, through p. 5, l. 6 (square-bracketed material in original) (emphasis added); see also id., p. 7, ll. 1-3 and 7-9. However, Appellants respectfully disagree.

The table containing the connection descriptor (which the Examiner attempts to read on the recited second level wrapper table) is not created "for *each instance*" of the patron object. Nor do the semantic links point to or reference an individual *instance* of the EosMapElement class, as alleged by the Examiner. Instead, as explained in Section B above, the table containing the connection descriptor contains a generic description of connections between *classes* of objects, and is applied to the class as a whole, rather than to individual instances of the class. See Mitchell et al., col. 21, ll. 41-47; and col. 22, ll. 25-31.

Further, as explained in section B above, Mitchell et al. fails to teach that the probe (which the Examiner attempts to read on the recited pointer in the first level wrapper table) references the table with the connection descriptor (which the Examiner attempts to read on the recited second level wrapper table). Rather, as shown in

FIG. 1, the probes point to mapper 103 (which the Examiner reads as instances of an association class). See Mitchell et al., FIG. 1; and col. 14, ll. 44-46.

For at least these reasons, the Examiner's rejection of claims 6, 26 and 42 under 35 U.S.C. § 102(b) is not supported by Mitchell et al. Accordingly, the rejection of these claims must be reversed and the claims allowed.

F. The rejection of claims 7, 8, 27, 28, 43 and 44 under 35 U.S.C. 102(b) must be reversed because Mitchell et al. fails to teach, *for each new class instance and corresponding association class instance,* creating a new first level wrapper table including a pointer to a new second level wrapper table associated with the association class and creating a pointer in the new second level wrapper table that references the new instance of the association class.

Claims 7, 8, 27, 28 43 and 44 each depend from one of claims 6, 26 and 42. As explained in Section E above, the Examiner's rejections of claims 6, 26 and 42 under 35 U.S.C. § 102(b) lacks support in Mitchell et al. Therefore, the Examiner's rejection of claims 7, 8, 27, 28 43 and 44 under 35 U.S.C. § 102(b) likewise lacks support in Mitchell et al. for at least the same reasons given above. Accordingly, the rejection of these claims must be reversed and the claims allowed.

Further, in the rejection of claims 7, 8, 27, 28 43 and 44, the Examiner asserts that Mitchell et al. teaches "new wrappers and pointers are created for new associations on new class *instances* as claimed." Final Office Action, p. 5, ll. 7-10 (emphasis added) (citing "the discussions regarding claims 4-6 above, and Column 21, line 34 - Column 23, line 48 of Mitchell's specification"). However, as explained in Section B above, the table containing the connection descriptor contains a generic description of connections between *classes* of objects, and is applied to the class as a whole, rather than to

individual instances of the class. See Mitchell et al., col. 21, ll. 41-47; and col. 22, ll. 25-31. The Examiner has pointed to no teaching in Mitchell et al. (and Appellants can find none), that the table containing the connection descriptor (which the Examiner attempts to read on the second level wrapper table) is created for each new class instance, or any teaching that this table contains a reference to newly created *instances* of the association class.

Consequently, the Examiner's rejection of claims 7, 8, 27, 28 43 and 44 under 35 U.S.C. § 102(b) lacks support in Mitchell et al. Accordingly, Appellants respectfully request that the rejection of these claims be reversed and the claims allowed.

G. The rejection of dependent claims 8, 28 and 44 under 35 U.S.C. 102(b) must be reversed because Mitchell et al. fails to teach, *for each new class instance and corresponding association class instance,* determining all instances of the association class that reference the new class instance, and creating a pointer in the new second level wrapper table for each instance of the association class.

In the rejection of claims 8, 28 and 44, the Examiner has not shown *or even alleged* that Mitchell et al. determines *all* mappers 103 (which the Examiner attempts to read on the recited instances of the association class) that reference a new class instance. As explained above, the table containing the connection descriptor contains a generic description of connections between classes of objects, rather than to connections between individual instances of the class, and is used to *initialize* the semantic links. See Mitchell et al., col. 21, ll. 41-47; and col. 22, ll. 25-31. Consequently, the "semantic link description" (which the Examiner attempts to read on the recited pointer in second table) does not reference an *instance* of the

EosMapElement class (which the Examiner attempts to read on the recited association class). Instead, it defines the class.

Therefore, the rejection of claims 8, 28 and 44 under 35 U.S.C. § 102(b) lacks support in Mitchell et al. Accordingly, Appellants respectfully request that the rejection of these claims be reversed and the claims allowed.

H. The rejection of claims 9 and 45 under 35 U.S.C. 102(b) must be reversed because Mitchell et al. fails to teach, *for each class instance associated with N association class classes,* creating a first level wrapper table including N pointers to N second level wrapper tables associated with the association classes.

Claims 9 and 45 are "rejected on the same basis as claims 6-8." See Final Office Action, p. 5, ll. 11-13; p. 7, ll. 7-9. However, the Examiner has not shown *or even alleged* that Mitchell et al. creates a first level wrapper table including N pointers to N second level wrapper tables associated with the association classes, for each class instance associated with N association class classes.

As explained in section B above, Mitchell et al. fails to teach that the probes (which the Examiner attempts to read on the N pointers to second level wrapper tables recited in claims 9 and 45) point to the table with the connection descriptor (which the Examiner attempts to read on the second level wrapper table recited in claims 9 and 45). Rather, as shown in FIG. 1, the probes point to mapper 103 (which the Examiner reads as instances of an association class). See Mitchell et al., FIG. 1; and col. 14, ll. 44-46.

For at least these reasons, Mitchell et al. does not support the Examiner's rejection of claims 9 and 45 under 35 U.S.C. § 102(b). Accordingly, Appellants

respectfully request that the rejection of these claims be reversed and the claims allowed.

I. The rejection of claims 10 and 46 under 35 U.S.C. 102(b) must be reversed because Mitchell et al. fails to teach, *for each class instance,* creating a first level wrapper table including a second pointer to a second level wrapper table associated with a second association class, and creating X pointers in the second level wrapper table that each reference an individual instance of the second association class.

In the rejection of claims 10 and 46, the Examiner asserts that Mitchell et al. discloses "X instances [e.g. 104, 107] of a second association class [EosFieldElement] that *each reference* [see Fig. 1] *the class instance*." Final Office Action, p. 5, ll. 19-21 (square-bracketed material in original) (emphasis added); see also id., p. 7, ll. 7-9. However, the right and left side type elements 104 and 107 of Mitchell et al. are bound to *different* patron objects 102 and 106. See Mitchell et al., FIG. 1. Consequently, these elements 104 and 107 do not "each reference an *individual instance* of the first [or second] association class," as recited in claims 10 and 46.

In addition, as explained in Section B above, Mitchell et al. fails to teach that the probes (which the Examiner attempts to read on the recited pointers to second level wrapper tables) point to the table with the connection descriptor (which the Examiner attempts to read on the recited second level wrapper table). Rather, as shown in FIG. 1, the probes point to mapper 103 (which the Examiner reads as instances of an association class). See Mitchell et al., FIG. 1; and col. 14, ll. 44-46.

For at least these additional reasons, Mitchell et al. does not support the Examiner's rejection of claims 10 and 46 under 35 U.S.C. § 102(b). Accordingly,

Appellants respectfully request that the rejection of these claims be reversed and the claims allowed.

J. The rejection of claims 11, 29 and 47 under 35 U.S.C. 102(b) must be reversed because Mitchell et al. fails to teach receiving an association traversal request for a class instance, and performing an association traversal process including (1) accessing a first table including a pointer to a second table, and (2) accessing the second table, using the pointer, to obtain pointers to each association instance that references the class instance.

With respect to claims 11, 29 and 47, the Examiner asserts that Mitchell et al. "discloses the method for performing association traversals as claimed. See Figures 1-3 and the corresponding portions of Mitchell's specification, as well as the discussions regarding claims 1-10 above." Final Office Action, p. 6, ll. 8-11; see also id., p. 7, ll. 4-9. Specifically, the Examiner asserts that Mitchell et al. teaches "receiving an association traversal request [due to update of or modification to a Patron Object] for a class instance [Patron Object (e.g. 102, 106)]; and performing an association traversal process ... [see Claims 1-5 above] as claimed." Id., p. 6, ll. 8-16 (square-bracketed material in original); see also id., p. 7, ll. 1-3 and 7-9. Appellants respectfully disagree with the Examiner's interpretation of Mitchell et al.

Mitchell et al. describes the action that takes place "due to update of or modification to a Patron Object" in column 14, lines 8-63. Specifically, changing one of the fields (e.g., field 105) of a Patron Object causes any probes set on that field to "fire," and mapper 103 "simply gets the values from the source side parameter [e.g., field 105] (that changed) and sets the value on the destination side [field 101] to this new value." Id., col. 14, ll. 34-38. However, Appellants can find no disclosure in Mitchell et al. (and

the Examiner can point to none) that would equate the "firing" of a probe "due to update of or modification to a Patron Object" (see Mitchell et al., col. 14, ll. 8-43) with "an association traversal request for a class instance."

Further, Mitchell et al. does not teach performing an association traversal in response to an "update of or modification to a Patron Object." As explained above, the firing of the probe simply triggers mapper 103 to synchronize the probed fields. Id.

Moreover, as explained in Section B above, Mitchell et al. fails to teach that the probe (which the Examiner attempts to read on the recited pointer in the first table) references the table with the connection descriptor (which the Examiner attempts to read on the recited second table). Rather, as shown in FIG. 1, the probes point to mapper 103 (which the Examiner reads as instances of an association class). See Mitchell et al., FIG. 1; and col. 14, ll. 44-46.

Finally, Mitchell et al. does not teach accessing the table containing the connection descriptor (which the Examiner attempts to read on the second table) to obtain pointers to each association class *instance* that references the class instance. As explained in Section B above, the table containing the connection descriptor contains "the semantic link *description*" that is used to *initialize* the semantic links. See Mitchell et al., col. 21, ll. 41-47. Consequently, the "semantic link description" (which the Examiner attempts to read on the recited pointer in second table) does not reference an *instance* of the EosMapElement class (which the Examiner attempts to read on the recited association class). Instead, the semantic link description is a generic description of connections between classes of objects, that is used to initialize the semantic links. Id.

For at least these reasons, the Examiner's rejection of claims 11, 29 and 47 under 35 U.S.C. § 102(b) is not supported by Mitchell et al. Accordingly, the rejection of these claims under 35 U.S.C. § 102(b) must be reversed and the claims allowed.

K. The rejection of claims 13, 31 and 49 under 35 U.S.C. 102(b) must be reversed for at least the same reasons given above with respect to claims 11, 29 and 47.

Claims 13, 31 and 49 depend from claims 11, 29 and 47, respectively. As explained in Section J above, the Examiner's rejection of claims 11, 29 and 47 under 35 U.S.C. § 102(b) is not supported by Mitchell et al. Therefore, the Examiner's rejection of claims 13, 31 and 49 under 35 U.S.C. § 102(b) likewise lacks support in Mitchell et al. for at least the same reasons given above. Accordingly, the rejection of these claims must be reversed and the claims allowed.

L. The rejection of claims 14-20, 32, 33 and 50-53 under 35 U.S.C. 102(b) must be reversed because Mitchell et al. fails to teach each and every recitation of these claims.

The Examiner asserts that "claims 14-20 [and claims 32, 33 and 50-53] repeat limitations of claims 1-11 in various combinations" and thus rejects these claims "on substantially the same basis as one or more of claims 1-11 above." See Final Office Action, p. 6, ll. 20-22, and p. 7, ll. 4-9. However, Appellants respectfully disagree with the Examiner's characterization of these claims. Contrary to the Examiner's assertions, each of claims 14-20, 32, 33 and 50-53 contain recitations not present in any of claims 1-11. Further, the Examiner has not shown *or even alleged* that Mitchell et al. teaches such recitations, and Mitchell et al. fails to anticipate any of claims 14-20, 32, 33 and 50-53, for at least the reasons given below.

1. The rejection of claims 14, 15, 32, 33, 50 and 51 under 35 U.S.C. 102(b) must be reversed because Mitchell et al. fails to teach receiving an association traversal request for a class instance, and obtaining pointers to each association instance that references the class instance from a wrapper table corresponding to the class instance.

The Examiner asserts that Mitchell et al. teaches "receiving an association traversal request [due to update of or modification to a Patron Object] for a class instance [Patron Object (e.g. 102, 106)]." Final Office Action, p. 6, ll. 8-16 (square-bracketed material in original); see also id., p. 7, ll. 1-3 and 7-9. However, as explained in Section J above, the "update of or modification to a Patron Object," *per se*, is not "an association traversal request for a class instance."

Further, Mitchell et al. does not teach "obtaining pointers to *each* association instance that references the class instance from a wrapper table corresponding to the class instance" in response to an "update of or modification to a Patron Object." As explained in Section B above, the update simply triggers mapper 103 to synchronize the individual probed *fields* of the Patron Object. See Mitchell et al., col. 14, ll. 8-63.

For at least these reasons, the Examiner's rejection of claims 14, 15, 32, 33, 50 and 51 under 35 U.S.C. § 102(b) is not supported by Mitchell et al. Accordingly, the rejection of these claims under 35 U.S.C. § 102(b) must be reversed and the claims allowed.

2. **The rejection of claims 16, 17, 52 and 53 under 35 U.S.C. 102(b) must be reversed because Mitchell et al. fails to teach a response including information reflecting a relationship between the selected class instance and other class instances that are referenced by the same association class instances, wherein the response is generated using pointers defined in a table associated with the selected class instance that reference the common association class instances.**

Mitchell et al. does not teach "a response" to an "update of or modification to a Patron Object" including "information reflecting a relationship between the selected class instance and other class instances that are referenced by the same association class instances." As explained in Section B above, modification of a Patron Object simply triggers mapper 103 to synchronize the individual probed fields of the Patron Object. See Mitchell et al., col. 14, ll. 8-63. "The mapper [103] is never referenced or visible to . . . the patron objects" (id. *col.* 9, ll. 15-16), and "the [patron] object has no knowledge of which objects (if any) have placed [the] probes" (id., col. 31, ll. 36-37).

For at least these reasons, the Examiner's rejection of claims 16 and 52 under 35 U.S.C. § 102(b) is not supported by Mitchell et al. Accordingly, the rejection of these claims under 35 U.S.C. § 102(b) must be reversed and the claims allowed.

3. **The rejection of claim 18 under 35 U.S.C. 102(b) must be reversed because Mitchell et al. fails to teach a client for generating a request for information reflecting the relationship between selected objects and a server including an object manager for processing the request using an object wrapper associated with one object of the selected objects.**

In the rejection of claim 18, the Examiner asserts that Mitchell et al. teaches a "request [due to update of or modification to a Patron Object]." Final Office Action, p. 6, ll. 8-16 (square-bracketed material in original). However, the Examiner has not shown

or even alleged that an update to an individual object equates to a request for information reflecting the relationship between selected *objects*.

Further, while the EosConnection class described by Mitchell et al. is described as a "wrapper" (id., col. 16, l. 11), it is not used to process "a request for information reflecting the relationship between selected objects." Instead, as explained in Section A above, it is "a simple wrapper around the RPC [Remote Procedure Call] calls used in the network connection." Mitchell et al., col. 16, ll. 11-12 (emphasis added). Therefore, Mitchell et al. fails to teach "an object manager for processing the request using an object *wrapper* associated with one object of the selected objects."

For at least these reasons, the Examiner's rejection of claim 18 under 35 U.S.C. § 102(b) is not supported by Mitchell et al. Accordingly, the rejection of this claim under 35 U.S.C. § 102(b) must be reversed and the claim allowed.

4. The rejection of claims 19 and 20 under 35 U.S.C. 102(b) must be reversed because Mitchell et al. for at least the same reasons given above with respect to claim 18.

Claims 19 and 20 depend from claim 18. As explained in Section L(3) above, the Examiner's rejection of claim 18 under 35 U.S.C. § 102(b) is not supported by Mitchell et al. Therefore, the Examiner's rejection of claims 19 and 20 under 35 U.S.C. § 102(b) likewise lacks support in Mitchell et al. for at least the same reasons given above. Accordingly, the rejection of these claims must be reversed and the claims allowed.

M. Conclusion

For at least the reasons given above, the Examiner's rejection of pending claims 1-2, 5-11, 13-22, 25-29, 31-33, 37, 38, 41-47 and 49-53 under 35 U.S.C.§ 102(b) lacks support in Mitchell et al. (U.S. Patent 5,872,973), the only reference relied upon in the rejections. Accordingly, Appellants respectfully request that the Examiner's rejections be reversed and the claims allowed.

To the extent any extension of time under 37 C.F.R. § 1.136 is required to obtain entry of this Appeal Brief, such extension is hereby respectfully requested. If there are any fees due under 37 C.F.R. §§ 1.16 or 1.17 which are not enclosed herewith, including any fees required for an extension of time under 37 C.F.R. § 1.136, please charge such fees to our Deposit Account No. 06-0916.

Respectfully submitted,

FINNEGAN, HENDERSON, FARABOW,
GARRETT & DUNNER, L.L.P.

Dated: October 5, 2005

By: _____
John M. Mulcahy
Reg. No. 55,940

VIII. CLAIMS APPENDIX

1. (Previously presented) A computer-implemented method for determining a relationship between objects related to a common information model, the objects including at least a first and second instance and an association that represents an instance of an association class, the method comprising:

creating, for the first instance, a reverse link that defines a relationship between the first instance and the association, wherein the instance is associated with a first wrapper defining the reverse link, wherein creating the reverse link includes:

defining a pointer in a first table of the first wrapper that references a second table; and

defining a pointer in the second table that references the instance of the association class; and

determining a relationship between the first and second instances based on the reverse link.

2. (Original) The method of claim 1, wherein each association reflects a relationship between a respective association and a corresponding associated object.

3-4. (Canceled).

5. (Previously presented) The method of claim 1, wherein determining a relationship includes:

collecting a reference reflecting a relationship between the association and the second instance based on the pointer in the second table.

6. (Previously presented) A method for maintaining reverse links in a object-oriented environment including class instances and associations, the method comprising:

for each class instance associated with N instances of an association class that each reference the class instance, wherein N represents an integer value greater than or equal to one:

(i) creating a first level wrapper table including a pointer to a second level wrapper table associated with the association class; and

(ii) creating N pointers in the second level wrapper table that each reference an individual instance of the association class.

7. (Previously presented) The method of claim 6, further comprising:

for each new class instance and corresponding new association class instance that reference the new class instance that is created:

(iii) creating a new first level wrapper table including a pointer to a new second level wrapper table associated with the association class; and

(iv) creating a pointer in the new second level wrapper table that references the new instance of the association class.

8. (Previously presented) The method of claim 6, further comprising:

for each new class instance and corresponding new association class instance that is created:

(iii) determining all instances of the association class that reference the new class instance;

(iv) creating a new first level wrapper table including a pointer to a new second level wrapper table associated with the association class; and

(v) creating a pointer in the new second level wrapper table for each instance of the association class determined in step (iii).

9. (Previously presented) A method for maintaining reverse links in an object-oriented environment including instances, association classes and association class instances, the method comprising:

for each class instance associated with N association class classes that each include at least one association class instance that references the class instance, wherein N represents an integer value greater than or equal to one:

creating a first level wrapper table including N pointers to N second level wrapper tables associated with the association classes.

10. (Previously presented) A method for maintaining reverse links in a object-oriented environment including class instances and associations, the method comprising:

for each class instance associated with N instances of a first association class that each reference the class instance, and X instances of a second association class that each reference the class instance, wherein N and X represent integer values greater than or equal to one:

(i) creating a first level wrapper table including:

a first pointer to a second level wrapper table associated with the first association class, and

a second pointer to a second level wrapper table associated with the second association class;

(ii) creating N pointers, in the second level wrapper table associated with the first association class, that each reference an individual instance of the first association class; and

(iii) creating X pointers, in the second level wrapper table associated with the second association class, that each reference an individual instance of the second association class.

11. (Previously presented) A method for performing association traversals in an object-oriented environment including a plurality of class instances and association instances, comprising:

receiving an association traversal request for a class instance; and

performing an association traversal process based on pointer information reflecting a relationship between the class instance and all association instances that reference the class instance, wherein the association traversal process includes:

accessing a first table including a pointer to a second table; and

accessing the second table, using the pointer, to obtain pointers to each association instance that references the class instance.

12. (Canceled).

13. (Previously presented) The method of claim 11, wherein the association traversal process further includes:

for each association instance pointed to by the second table:

collecting a reference to another class instance that is referenced by the association instance.

14. (Previously presented) A method for performing association traversals in an object-oriented environment including a plurality of class instances and association instances, comprising:

receiving an association traversal request for a class instance;

obtaining pointers to each association instance that references the class instance from a wrapper table corresponding to the class instance; and

collecting, from the wrapper table, references to other class instances that are referenced by the association instance.

15. (Previously presented) A method for performing association traversals in an object-oriented environment including a plurality of class instances and association instances, comprising:

receiving an association traversal request for a class instance; and

obtaining pointers to each association instance that references the class instance from a wrapper table corresponding to the class instance.

16. (Previously presented) In a system comprising a client and a server, a method for performing association traversals performed by the client, comprising:

generating a request for relationship information associated with a selected class instance; and

receiving a response including information reflecting a relationship between the selected class instance and other class instances that are referenced by the same association class instances, wherein the response is generated using pointers defined in a table associated with the selected class instance that reference the common association class instances.

17. (Original) In a system comprising a client and a server, a method for performing association traversals performed by the server, comprising;

receiving a request for relationship information associated with a selected class instance; and

generating a response including information reflecting a relationship between the selected class instance and other class instances that are referenced by the same association class instances, based on pointers defined in a table associated with the selected class instance that reference the common association class instances.

18. (Original) A system for performing association traversals, comprising:

a client for generating a request for information reflecting the relationship between selected objects defined in a repository; and

a server including:

the repository for storing objects, wherein a set of the objects are associated with object wrappers, and

an object manager for processing the request by using an object wrapper associated with one object of the selected objects.

19. (Original) The system of claim 18, wherein the object wrapper associated with one object includes pointers to all association instances that reference the one object.

20. (Original) The system of claim 19, wherein the object manager uses the pointers to collect references to the selected objects.

21. (Previously presented) A system for traversing associations in a common information model implemented environment, the model comprising at least a first and second instance and an association that represents an instance of an association class, comprising:

means for creating, for the first instance, a reverse link that defines a relationship between the first instance and the association, wherein the instance is associated with a first wrapper defining the reverse link, wherein the means for creating the reverse link includes:

means for defining a pointer in a first table of the first wrapper that references a second table; and

means for defining a pointer in the second table that references the instance of the association class; and

means for determining a relationship between the first and second instances based on the reverse link.

22. (Original) The system of claim 21, wherein each association reflects a relationship between a respective association and a corresponding associated object.

23-24. (Canceled).

25. (Previously presented) The system of claim 21, wherein the means for determining a relationship includes:

means for collecting a reference reflecting a relationship between the association and the second instance based on the pointer in the second table.

26. (Previously presented) A system for maintaining reverse links in a object-oriented environment including class instances and associations, comprising:

means for creating, for each class instance associated with N instances of an association class that each reference the class instance, wherein N represents an integer value greater than or equal to one, a first level wrapper table including a pointer to a second level wrapper table associated with the association class; and

means for creating N pointers in the second level wrapper table that each point to an individual instance of the association class.

27. (Previously presented) The system of claim 26, further comprising:

means for creating, for each new class instance and corresponding new association class instance that reference the new class instance that is created, a new first level wrapper table including a pointer to a new second level wrapper table associated with the association class; and

means for creating a pointer in the new second level wrapper table that references the new instance of the association class.

28. (Previously presented) The system of claim 26, further comprising:

means for determining, for each new class instance and corresponding new association class instance that is created, all instances of the association class that reference the new class instance;

means for creating a new first level wrapper table including a pointer to a new second level wrapper table associated with the association class; and

means for creating a pointer in the new second level wrapper table for each instance of the association class determined by the means for determining.

29. (Previously presented) A system for performing association traversals in an object-oriented environment including a plurality of class instances and association instances, comprising:

means for receiving an association traversal request for a class instance; and

means for performing an association traversal process based on pointer information reflecting a relationship between the class instance and all association instances that reference the class instance, wherein the means for performing an association traversal process includes:

means for accessing a first table including a pointer to a second table; and

means for accessing the second table, using the pointer, to obtain pointers to each association instance that references the class instance.

30. (Canceled).

31. (Previously presented) The system of claim 29, wherein the means for performing an association traversal process further includes:

means for collecting, for each association instance pointed to by the second table, a reference to another class instance that is referenced by the association instance.

32. (Previously presented) A system for performing association traversals in an object-oriented environment including a plurality of class instances and association instances, comprising:

means for receiving an association traversal request for a class instance;

means for obtaining pointers to each association instance that references the class instance from a wrapper table corresponding to the class instance; and

means for collecting, from the wrapper table, references to other class instances that are referenced by the association instance.

33. (Previously presented) A system for performing association traversals in an object-oriented environment including a plurality of class instances and association instances, comprising:

means for receiving an association traversal request for a class instance; and

means for obtaining pointers to each association instance that references the class instance from a wrapper table corresponding to the class instance.

34-36. (Canceled).

37. (Previously presented) A computer-readable medium including instructions for performing a method, when executed by a processor, for determining a relationship between objects related to a common information model, the objects including at least a first and second instance and an association that represents an instance of an association class, the method comprising:

creating, for the first instance, a reverse link that defines a relationship between the first instance and the association, wherein the instance is associated with a first wrapper defining the reverse link, and wherein creating the reverse link includes:

defining a pointer in a first table of the first wrapper that references a second table; and

defining a pointer in the second table that references the instance of the association class; and

determining a relationship between the first and second instances based on the reverse link.

38. (Original) The computer-readable medium of claim 37, wherein each association reflects a relationship between a respective association and a corresponding associated object.

39-40. (Canceled).

41. (Previously presented) The computer-readable medium of claim 37, wherein the step of determining a relationship includes:

collecting a reference reflecting a relationship between the association and the second instance based on the pointer in the second table.

42. (Previously presented) A computer-readable medium including instructions for performing a method, when executed by a processor, for maintaining reverse links in a object-oriented environment including class instances and associations, the method comprising:

for each class instance associated with N instances of an association class that each reference the class instance, wherein N represents an integer value greater than or equal to one:

(i) creating a first level wrapper table including a pointer to a second level wrapper table associated with the association class; and

(ii) creating N pointers in the second level wrapper table that each reference an individual instance of the association class.

43. (Previously presented) The computer-readable medium of claim 42, wherein the method further comprises:

for each new class instance and corresponding new association class instance that reference the new class instance that is created:

(iii) creating a new first level wrapper table including a pointer to a new second level wrapper table associated with the association class; and

(iv) creating a pointer in the new second level wrapper table that references the new instance of the association class.

44. (Previously presented) The computer-readable medium of claim 42, wherein the method further comprises:

for each new class instance and corresponding new association class instance that is created:

(iii) determining all instances of the association class that reference the new class instance;

(iv) creating a new first level wrapper table including a pointer to a new second level wrapper table associated with the association class; and

(v) creating a pointer in the new second level wrapper table for each instance of the association class determined in step (iii).

45. (Previously presented) A computer-readable medium including instructions for performing a method, when executed by a processor, for maintaining reverse links in a object-oriented environment including class instances and associations, the method comprising:

for each class instance associated with N association classes that each include at least one association class instance that references the class instance, wherein N represents an integer value greater than or equal to one:

creating a first level wrapper table including N pointers to N second level wrapper tables associated with the association classes.

46. (Previously presented) A computer-readable medium including instructions for performing a method, when executed by a processor, for maintaining reverse links in a object-oriented environment including class instances and associations, the method comprising:

for each class instance associated with N instances of a first association class that each reference the class instance, and X instances of a second association class that each reference the class instance, wherein N and X represent integer values greater than or equal to one:

(i) creating a first level wrapper table including:

a first pointer to a second level wrapper table associated with the first association class, and

a second pointer to a second level wrapper table associated with the second association class;

(ii) creating N pointers, in the second level wrapper table associated with the first association class, that each reference an individual instance of the first association class; and

(iii) creating X pointers, in the second level wrapper table associated with the second association class, that each reference an individual instance of the second association class.

47. (Previously presented) A computer-readable medium including instructions for performing a method, when executed by a processor, for performing association traversals in an object-oriented environment including a plurality of class instances and association instances, comprising:

receiving an association traversal request for a class instance; and

performing an association traversal process based on pointer information reflecting a relationship between the class instance and all association instances that reference the class instance, wherein the association traversal process includes:

accessing a first table including a pointer to a second table; and

accessing the second table, using the pointer, to obtain pointers to each association instance that references the class instance.

48. (Canceled).

49. (Previously presented) The computer-readable medium of claim 47, wherein the association traversal process further includes:

for each association instance pointed to by the second table:

collecting a reference to another class instance that is referenced by the association instance.

50. (Previously presented) A computer-readable medium including instructions for performing a method, when executed by a processor, for performing association traversals in an object-oriented environment including a plurality of class instances and association instances, comprising:

receiving an association traversal request for a class instance;

obtaining pointers to each association instance that references the class instance from a wrapper table corresponding to the class instance; and

collecting, from the wrapper table, references to other class instances that are referenced by the association instance.

51. (Previously presented) A computer-readable medium including instructions for performing a method, when executed by a processor, for performing association traversals in an object-oriented environment including a plurality of class instances and association instances, comprising:

receiving an association traversal request for a class instance; and

obtaining pointers to each association instance that references the class instance from a wrapper table corresponding to the class instance.

52. (Previously presented) In a system comprising a client and a server, a computer-readable medium including instructions for performing a method, when executed by a processor, for performing association traversals performed by the client, comprising:

generating a request for relationship information associated with a selected class instance;

receiving a response including information reflecting a relationship between the selected class instance and other class instances that are referenced by the same association class instances, wherein the response is generated using pointers defined in a table associated with the selected class instance that reference the common association class instances.

53. (Original) In a system comprising a client and a server, a computer-readable medium including instructions for performing a method, when executed by a processor, for performing association traversals performed by the server, comprising;

receiving a request for relationship information associated with a selected class instance;

generating a response including information reflecting a relationship between the selected class instance and other class instances that are referenced by the same association class instances, based on pointers defined in a table associated with the selected class instance that reference the common association class instances.

54-56. (Canceled).

IX. EVIDENCE APPENDIX

None.

X. RELATED PROCEEDINGS APPENDIX

None.